We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

8

Transformations of Image Filters for Machine Vision Using Complex-Valued Neural Networks

Takehiko Ogawa
Takushoku University
Japan

1. Introduction

Neural networks expanded to complex domains have recently been studied in the field of computational intelligence. Complex-valued neural networks are effective for learning the relationships between complex inputs and outputs, and applications to complex analysis and complex image processing have been studied (Hirose, 2006). In addition, the effectiveness of the computational complexity and the number of training data has been confirmed when learning mappings in two-dimensional space (Nitta, 1997). Also, a method for complex-valued network inversion to produce an inverse mapping was proposed as a related technique using a complex-valued neural network (Ogawa, 2009). We can obtain forward mappings and inverse mappings in complex domains using these methods.

Image filters are important in bio-inspired systems, machine vision, and image processing. We can extract relevant information or remove unnecessary information from any given image using various image filters (Gonzalez & Woods, 2010; Pratt, 2007). In machine vision, image filters are effective for transforming, mapping, and making a required filter adaptive. In this work, we propose to adaptively map an image filter using a neural network to make an appropriate filter. We show that the mapping requires less training data when using complex-valued neural networks.

In this work, we examine the transformation of image filters using complex-valued neural networks. First, we conduct a simulation to learn the transformation, and demonstrate the capacity for forward and inverse mapping using complex-valued neural networks. We then demonstrate various transformations of image filters, such as Gaussian filters, using complex-valued neural networks. In addition, we present the results of image filtering using the transformed image filters.

2. Complex-valued neural networks

Complex-valued neural networks have recently been used to directly learn and recognize data in a complex region. These networks learn complex input–output relationships using complex weights and complex neurons. Various models have been proposed for representing these networks, such as the multilayer-type neural network (Benvenuto & Piazza, 1992), the self-organizing map (Hirose & Hara, 2003), and associative memory (Nemoto & Kubono, 1996), and a number of applications of these networks have also been studied.

Complex-valued neural networks are effective for processing data in a coordinate system where the phase rotates or for learning relationships in the frequency domain. Applications of complex-valued neural networks include adaptive design of patch antennas (Du et al., 2002), radar image processing (Hara & Hirose, 2004), and traffic-dependent optimal control of traffic signals (Nishikawa & Kuroe, 2004).

2.1 Complex-valued multilayer neural networks

In this study, complex-valued multilayer neural networks are used for filter transformation based on their capacity for mapping. Complex-valued multilayer neural networks are an extension of the usual multilayer neural networks to complex regions. This method determines the relationships between complex inputs and outputs using complex neurons and complex weights. They are typically composed of an input layer, some hidden layers, and the output layers. All functions can be realized in complex-valued multilayer neural networks if there is at least one hidden layer and a normal multilayer network. In this study, we used a neural network with three layers, namely, input layer, hidden layer, and output layer, for simulation.

In this study, we considered a multilayer neural network based on an error backpropagation learning method. This model learns complex input–output relationships using complex weights and complex neurons. Complex-valued neural networks are classified on the basis of their architecture and the type of neurons found in these networks. For instance, one type of complex-valued neural networks is based on the transfer function of the neuron, while another consists of real-type neurons. Here, we consider a neuron that independently applies a sigmoid function to the real and imaginary parts of the weighted sum of inputs. This neuron independently applies a complex sigmoid function to each real part and imaginary part, which can be defined as

$$f_C(s) = f(s_R) + if(s_I), \quad f(u) = \frac{1 - e^{-u}}{1 + e^{-u}} \tag{1}$$

where i and $s = s_R + is_I$ indicate the imaginary unit and the weighted sum of the neuron input, respectively. The architecture of the complex-valued sigmoid neuron is shown in Fig. 1. The neuron transforms the weighted sum $s = s_R + is_I$ of the input $x_n = x_{nR} + ix_{nI}$ and the weight $w_n = w_{nR} + iw_{nI}$ to the output $f_c(s)=f(s_R)+if(s_I)$ using the sigmoid function of equation (1). In this network, complex-valued neurons are used for the hidden layer and the output layer.

Complex-valued multilayer neural networks are usually used in two phases of learning and estimation, as shown in Fig. 2. In the learning phase, we provide the training input and output, and model the forward relation using the error backpropagation algorithm. During the estimation phase, we obtain the output for a given input by fixing the weights obtained in the leaning phase.

Next, we explain learning in a complex-valued neural network using the error backpropagation algorithm, which is enhanced for complex regions. Here, we consider a three-layer network with an input layer, a hidden layer, and an output layer. The output error $E = E_R + iE_I$ is defined by the squared error as

$$E_R = \frac{1}{2}\sum_r \left(y'_{rR} - y_{rR}\right)^2, \quad E_I = \frac{1}{2}\sum_r \left(y'_{rI} - y_{rI}\right)^2 \tag{2}$$

where $y'_r = y'_{rR} + iy'_{rI}$ and $y_r = y_{rR} + iy_{rI}$ are the r-th tutorial output and the network output, respectively. First, we formulate the weight update procedure between the hidden and output layers. The error signal $\delta_r = \delta_{rR} + i\delta_{rI}$ from the output layer is calculated by

$$\delta_{rR} = \left(y'_{rR} - y_{rR}\right)\left(1 - y_{rR}\right)\left(1 + y_{rR}\right), \quad \delta_{rI} = \left(y'_{rI} - y_{rI}\right)\left(1 - y_{rI}\right)\left(1 + y_{rI}\right). \tag{3}$$

Also, the gradient of the output error for the weight $w_{rk} = w_{rkR} + iw_{rkI}$ between the hidden and output layer is expressed by

$$\frac{\partial E_R}{\partial w_{rkR}} = \delta_{rR} v_{kR}, \quad \frac{\partial E_I}{\partial w_{rkR}} = \delta_{rI} v_{kI}, \quad \frac{\partial E_R}{\partial w_{rkI}} = -\delta_{rR} v_{kI}, \quad \frac{\partial E_I}{\partial w_{rkI}} = \delta_{rI} v_{kR} \tag{4}$$

where $v_k = v_{kR} + iv_{kI}$ indicates the input from the k-th hidden neuron. Therefore, the weights are updated by

$$w_{rkR}^{new} = w_{rkR}^{old} - \varepsilon_t \left(\delta_{rR} v_{kR} + \delta_{rI} v_{kI}\right), \quad w_{rkI}^{new} = w_{rkI}^{old} - \varepsilon_t \left(\delta_{rI} v_{kR} - \delta_{rR} v_{kI}\right) \tag{5}$$

where ε_t denotes a training gain. In this way, complex weights are updated between the hidden and output layers. Next, we formulate the weight update procedure between the input and hidden layers. The error signal $\delta_k = \delta_{kR} + i\delta_{kI}$ from the hidden layer is calculated by

$$\begin{aligned} \delta_{kR} &= \left(1 - v_{kR}\right)\left(1 - v_{kR}\right)\sum_r \left(\delta_{rR} w_{rkR} + \delta_{rI} w_{rkI}\right), \\ \delta_{kI} &= \left(1 - v_{kI}\right)\left(1 - v_{kI}\right)\sum_r \left(\delta_{rI} w_{rkR} - \delta_{rR} w_{rkI}\right) \end{aligned} \tag{6}$$

Also, the gradient of the output error for the weight $w_{km} = w_{kmR} + iw_{kmI}$ between the input and hidden layer is expressed by

$$\frac{\partial E_R}{\partial w_{kmR}} = \delta_{kR} x_{mR}, \quad \frac{\partial E_I}{\partial w_{kmR}} = \delta_{kI} x_{mI}, \quad \frac{\partial E_R}{\partial w_{kmI}} = -\delta_{kR} x_{mI}, \quad \frac{\partial E_I}{\partial w_{kmI}} = \delta_{kI} x_{mR} \tag{7}$$

where $x_m = x_{mR} + ix_{mI}$ indicates the input from the m-th input neuron. Therefore, the weights are updated by

$$w_{kmR}^{new} = w_{kmR}^{old} - \varepsilon_t \left(\delta_{kR} x_{mR} + \delta_{kI} x_{mI}\right), \quad w_{kmI}^{new} = w_{kmI}^{old} - \varepsilon_t \left(\delta_{kI} x_{mR} - \delta_{kR} x_{mI}\right) \tag{8}$$

where ε_t is a training gain. In this way, the complex weights between the input and hidden layers are updated. The input-output relationship is learned by correcting each complex weight according to the above equations. In general, the afore-mentioned weight correction is repeated until a prescribed error value or repetition number. The principle of weight correction is based on the output error, as shown in Fig. 3.

The output is estimated from a given input using the learned complex-valued multilayer neural network. In the network studied, the output corresponding to the given input can be

estimated by fixing the weights obtained during learning with a given input and obtaining the output.

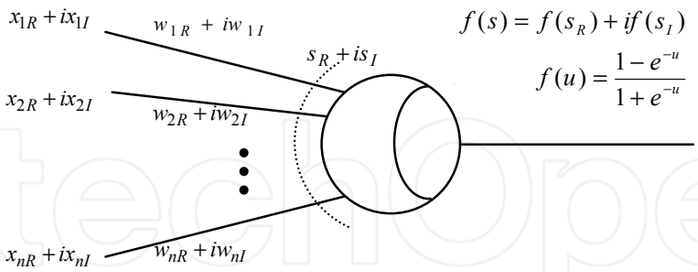


Fig. 1. A complex-valued sigmoid neuron.

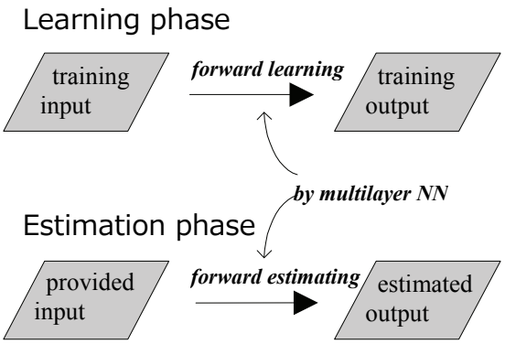


Fig. 2. Two-step estimation with complex-valued neural network.

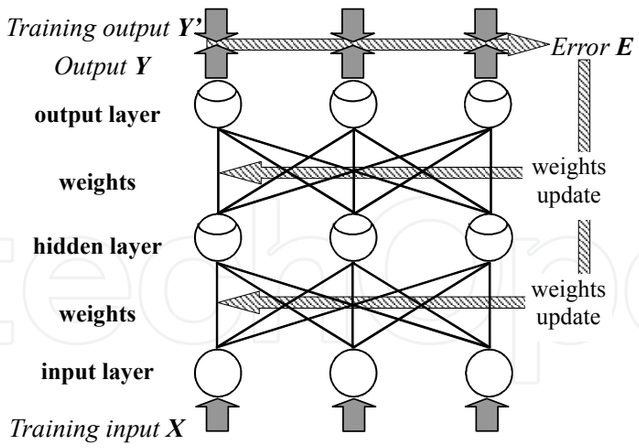


Fig. 3. Iterative correction of weights by error backpropagation learning.

2.2 Complex-valued network inversion

The inverse problem determines the inner mechanism or cause of an observed phenomenon. The concept of the inverse problem is shown in Fig. 4. The cause is estimated from a fixed model and a given result in the inverse problem, whereas the

result is determined from a given cause by using a certain fixed mathematical model in the forward problem (Groetsch, 1993). We consider the estimation process with the neural network from the viewpoint of the direction of the problem. A multilayer neural network is a normal solution for the forward problem because it estimates the output from the input based on the forward relationship obtained by training. In contrast, an inverse problem can be solved using a multilayer neural network inversely with the forward relationship obtained during training.

A normal multilayer neural network can be used to solve the forward problem. Given a normal multilayer network where training is completed, the input-output relationship is given by $y = f(w, x)$, where x, y, and f are the input vector, the output vector, and the function defined by the interlayer weights w of the network, respectively. Given the input vector x, the network calculates the output vector y. Linden and Kindermann proposed a method for network inversion (Linden & Kindermann, 1989). Using this method, we can determine the observed output data y with f fixed after finding the forward relationship f by training. Then, the input x can be updated according to the calculated input correction signal, based on the duality of the weights and input. The input is actually estimated from the output by the iterative updating of the input based on the output error. In this way, the inverse problem for estimating input x from output y is solved with a multilayer neural network by using the forward relationship inversely. Further, network inversion has been applied to image restoration (Valova et al., 1995) and the inverse kinematics of robot arms (Ogawa & Kanada, 2010).

The network is used in two phases, forward training and inverse estimation, to solve the inverse problem by network inversion, as shown in Fig. 5. During the training phase, we provide the training input x and the training output y and calculate the output error E. Then, the weight w is updated by

$$w^{new} = w^{old} - \varepsilon_t \frac{\partial E}{\partial w} \tag{9}$$

where ε_t is the training gain, because the output error is due to maladjustments of the weights. The forward relationship is obtained by repeating this update procedure. This procedure is based on the usual backpropagation method. During the inverse estimation phase, we fixed the relationship obtained during training, given the random input x and the test output y, and we calculated the output error E. The input x is then updated by

$$x^{new} = x^{old} - \varepsilon_e \frac{\partial E}{\partial x} \tag{10}$$

where ε_e denotes the input update gain, because the output error is due to the error of the input. The input is estimated from the output by repeating this update procedure. The principle of input correction is shown in Fig. 6.

A complex-valued network inversion is an extension of the principle of the network inversion to a complex number. In a complex-valued network inversion, the network inversion technique extended to a complex number is applied to a learned complex-valued

multilayer neural network to solve inverse problems. As a result, the complex input is estimated from a complex output, which is given to a learned complex-valued neural network. The network learns the input and output relationship using the error backpropagation algorithm extended to the complex region, which is explained in the previous section.

During the inverse estimation phase, the input is estimated from the given output. Thus, the provided initial random input is repeatedly updated by the output error, which is backpropagated to the input via the fixed weights. To provide an initial random input $x_m = x_{mR} + ix_{mI}$, the squared error $E = E_R + iE_I$ is calculated as

$$E_R = \frac{1}{2}\sum_r \left(y'_{rR} - y_{rR}\right)^2, \quad E_R = \frac{1}{2}\sum_r \left(y'_{rI} - y_{rI}\right)^2, \tag{11}$$

where $y_r = y_{rR} + iy_{rI}$ and $y'_r = y'_{rR} + iy'_{rI}$ indicate the network output and tutorial output, respectively. The error signals from the output and hidden layers are also formulated as

$$\delta_{rR} = \left(1 - y_{rR}\right)\cdot\left(1 + y_{rR}\right)\cdot\left(y'_{rR} - y_{rR}\right), \quad \delta_{rI} = \left(1 - y_{rI}\right)\cdot\left(1 + y_{rI}\right)\cdot\left(y'_{rI} - y_{rI}\right), \tag{12}$$

and

$$\begin{aligned} \delta_{kR} &= \left(1 - v_{kR}\right)\cdot\left(1 + v_{kR}\right)\cdot\sum_r \left(\delta_{rR} w_{rkR} - \delta_{rI} w_{rkI}\right), \\ \delta_{kI} &= \left(1 - v_{kI}\right)\cdot\left(1 + v_{kI}\right)\cdot\sum_r \left(\delta_{rI} w_{rkR} - \delta_{rR} w_{rkI}\right), \end{aligned} \tag{13}$$

where $v_k = v_{kR} + iv_{kI}$ indicates the input from the k-th hidden neuron to the output neurons. Equations (12) and (13) are similar to equations (3) and (6), respectively. The error signal to the input layer is then calculated by

$$\begin{aligned} \delta_{mR} &= \left(1 - x_{mR}\right)\cdot\left(1 + x_{mR}\right)\cdot\sum_k \left(\delta_{kR} w_{kmR} - \delta_{kI} w_{kmI}\right), \\ \delta_{mI} &= \left(1 - x_{mI}\right)\cdot\left(1 + x_{mI}\right)\cdot\sum_k \left(\delta_{kI} w_{kmR} - \delta_{kR} w_{kmI}\right), \end{aligned} \tag{14}$$

where $x_m = x_{mR} + ix_{mI}$ indicates the input from the m-th input neuron to the hidden neurons. The error signal $\delta_m = \delta_{mR} + i\delta_{mI}$ indicates a demand for input correction to the m-th input neuron, so the complex inputs are iteratively corrected as

$$x_{mR}^{new} = x_{mR}^{old} - \varepsilon_e \delta_{mR}, \quad x_{mI}^{new} = x_{mI}^{old} - \varepsilon_e \delta_{mI} \tag{15}$$

where ε_e is the inverse estimation gain. When the error reaches the target, the input correction is terminated and the obtained complex input becomes a solution. As a result, the complex input can be inversely estimated from the complex output by using the complex weight distribution obtained during training. This is similar to correcting the weights or the input iteratively during training and inverse estimation. However, the inverse estimation uses iterative corrections for a given pattern, which differs from training by repeated correction of plural patterns.

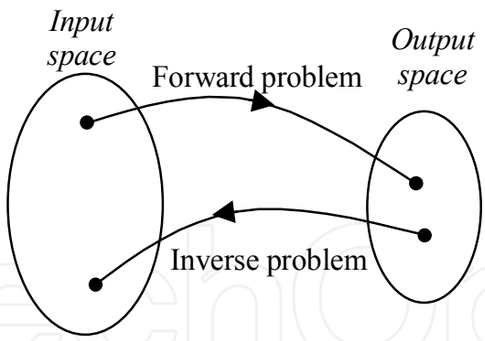


Fig. 4. Concept of inverse problems.

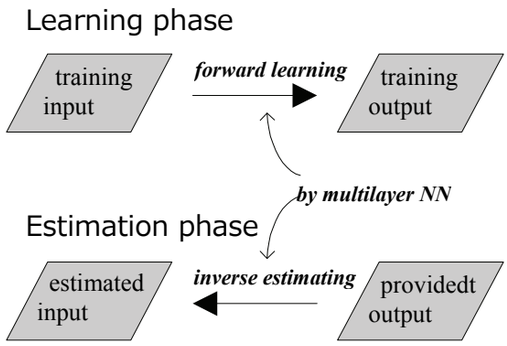


Fig. 5. Two-step estimation to solve inverse problems by complex-valued network inversion.

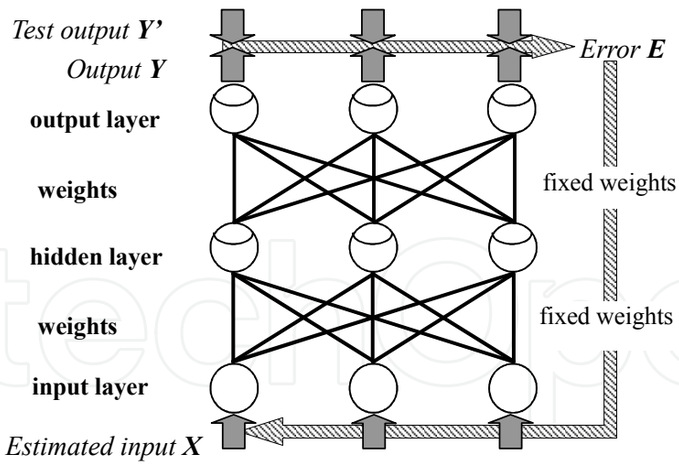


Fig. 6. Iterative correction of inputs by network inversion.

2.3 Learning of complex-mapping by complex-valued neural networks

A multilayer neural network can learn the nonlinear mapping relationships between inputs and outputs. A complex-valued multilayer neural network and a complex-valued network inversion can estimate, respectively, the forward mapping and the inverse mapping between an input and output. In this study, we examined the forward and inverse

estimation problems using these networks. In the complex-mapping problem, it is important that learning is related to the mapping of points on a complex plane. A complex-valued neural network has the particular advantage of learning a map with a coordinate system that rotates. We consider the use of complex-valued neural networks by replacing the mapping with a transformation of the rotation with a mapping on a complex plane.

We consider the problem of mapping a point on a complex plane using a network with an input and an output. The network learns the input/output relationship using a given complex learning pattern for input and output learning. During the forward estimation of a complex-valued multilayer neural network, we provide the input to obtain the output by fixing the weights obtained in learning. During the inverse estimation of the complex-valued network inversion, we provide a complex random input pattern and iteratively correct the input from a given complex output pattern by fixing the weights obtained in learning.

Various kinds of information can be expressed by assigning meanings to the coordinates in complex mapping problems. In this study, we examined the allocation of the values of image filters to a complex plane and performed various conversions. We can generally implement filters that achieve non-symmetric filtering and direction-related filtering. These are effective tools in machine vision. In this study, we considered the conversion of an image filter using complex-valued neural networks.

We examined the linear transformation of expansion/compression and rotation, and general projective conversion, which are important methods for transforming image filters. First, we considered a figure $g(x, y)$ in two-dimensional space, where each value of the coordinates (x, y) is moved to coordinates (x', y') repeatedly. Generally, a linear transform is expressed by

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{16}$$

using a matrix expression. Various linear transforms can be executed by choosing the four parameters a, b, c, and d. Here, we set the parameters $b = c = 0$ as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{17}$$

to consider the transform of expansion/compression shown in Fig. 7 (a). The parameters a and d indicate the rate of expansion/compression in the x-direction and y-direction, respectively. Expansion is indicated if these parameters are larger than 1.0, whereas compression is indicated if they are lower than 1.0. The transform of rotation is given as

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{18}$$

where θ denotes the counterclockwise angle around the origin, as shown in Fig. 7 (b). These are linear transforms, and they are realized by simple parameter estimation; however, we need to learn and estimate these transforms to illustrate the procedure for complex-valued

neural networks used in this study. More general geometric transforms include a projective transform. A projective transform is expressed by

$$x' = \frac{a_{11}x + a_{12}y + a_{13}}{a_{31}x + a_{32}y + a_{33}},\ y' = \frac{a_{21}x + a_{22}y + a_{23}}{a_{31}x + a_{32}y + a_{33}}, \tag{19}$$

which includes nine parameters a_{ij}. An example of a projective transform is shown in Fig. 7 (c). In this study, we consider these three transforms as geometric transforms on a complex plane, and we use them for estimation with complex-valued neural networks.

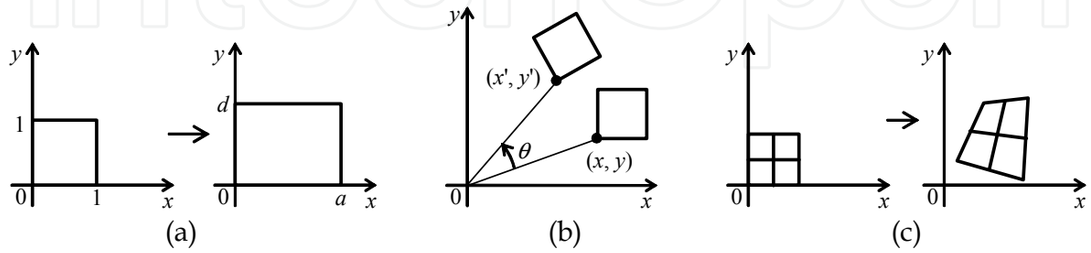


Fig. 7. Examples of geometric transforms: (a) expansion and compression, (b) rotation, and (c) projective transforms.

3. Image filters

Image filters are an important tool in the fields of bio-inspired systems, robot vision, and image processing. Relevant information is extracted or unnecessary information is removed from a given image with various image filters. In robot vision, it is useful if the image filter can transform or adaptively map when making a necessary filter. In this section, we propose to adaptively map an image filter using a neural network to make an appropriate filter. Complex-valued neural networks are known to effectively map using less training data.

A Gaussian function is often used as an expression of the frequency domain of a filter. For example, let $H(u)$ denote the 1-D frequency domain Gaussian filter as

$$H(u) = Ae^{-u^2/2\sigma^2} \tag{20}$$

where σ is the standard deviation of the Gaussian function. This function can be used as a one-dimensional low-pass filter. Moreover, let $H(u, v)$ denote the 2-D frequency domain Gaussian filter as

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2} \tag{21}$$

where $D(u,v) = \left(u^2 + v^2\right)$ and the origin, $u = v = 0$, is considered as the center of the filter.

Various filters can be made by applying the Gaussian function using the above-mentioned function, which basically provides a low-pass filter. For instance, a high-pass filter is produced by assuming $1 - H(u,v)$. Moreover, a filter of the lateral inhibition type can be produced based on the difference of two Gaussian functions with different standard deviations. This is called the DOG function and it can be used as an edge enhancement filter.

Moreover, an orientation selectivity filter can be produced from the rotated elliptic function by compressing and rotating a Gaussian function. This filter can extract a line segment or filter an image in a specific direction. A filter with various features can be produced by variously compressing and rotating the Gaussian filter. Thus, linear transformations and nonlinear transformations can be produced by learning using neural networks. As an example, we consider a nonlinear conversion, such as a projective transform. Fig. 8 shows examples of the plot of 2-D filter functions related to Gaussian functions: a low-pass filter, a high-pass filter, a lateral inhibition filter based on the difference of Gaussian functions, and an orientation selectivity filter that is obtained using an elliptical Gaussian function.

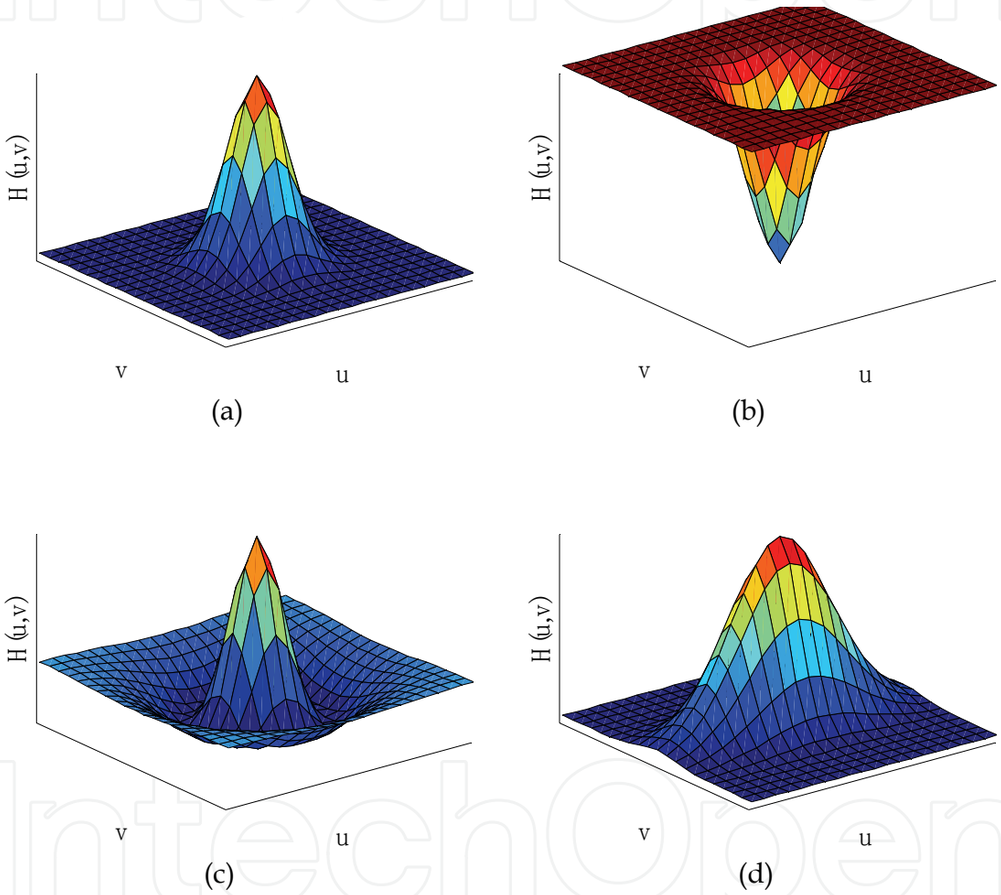


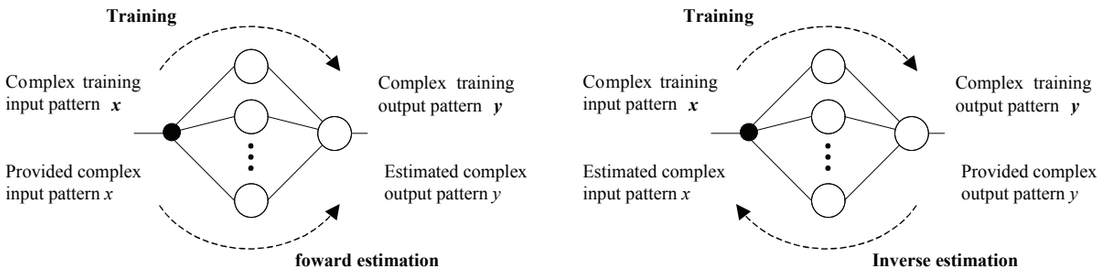
Fig. 8. Plots of 2-D filter functions related to Gaussian functions: (a) low-pass filter, (b) high-pass filter, (c) lateral inhibition filter based on the difference of Gaussian functions, and (d) a directional filter obtained using an elliptical Gaussian function.

4. Simulation

We conducted a simulation of a forward and inverse transform of the filter using forward and inverse mappings with a complex-valued multilayer neural network and a complex-valued network inversion. First, we examined the geometric conversions namely, the expansion/compression, rotation, and projective conversions, on a complex plane to demonstrate the learning procedure and the estimation of the complex mapping. The result

was a transformation of the geometric figure, the movement vector on the complex plane, and a conversion of the filter. In addition, we filtered the image with the transformed filter and showed the effect of the filter conversion with complex-valued neural networks.

We use a network with an input and an output, and we considered the problem of mapping one point to another on a complex plane. We provided complex input and output patterns and made the network learn the input/output relationship during learning. Then, we examined the forward estimation of the multilayer neural network and the inverse estimation using the network inversion of the learned network. Thus, we obtained an output from a given input by using the learned relation as it was and obtained the input from the given output by using the learned relation inversely. The network parameters and network architecture are presented in Table 1 and Fig. 9, respectively. The following section shows the results of the expansion/compression conversion, the rotational transformation, and the projective conversion.



(a) (b)

Fig. 9. Network architectures for (a) forward estimation and (b) inverse estimation.

	Expansion/compression		Rotation	Projective transform
	(symmetry)	(asymmetry)		
Input neurons	1			
Hidden neurons	10			
Output neurons	1			
Training rate ε_t	0.01			
Input correcting rate ε_e	0.1			
Maximum number of training epochs	50000	10000	50000	50000
Maximum number of estimation epochs	10000	10000	10000	10000
Criterion for learning convergence	0.0001	0.0001	0.0001	0.0001
Criterion for estimation convergence	0.001	0.001	0.001	0.001

Table 1. Network parameters.

4.1 Expansion/compression transform

We examined the learning, forward estimation, and inverse estimation of the expansion/compression transform on a complex plane with the complex-valued neural networks. Both symmetric and non-symmetric transforms along the vertical and horizontal axes are important for transforming an image filter such as the orientation selectivity filter.

Thus, symmetric transforms and non-symmetric transforms are examined here. As the learning data, we provided 22 pairs of data that satisfied $x' = ax$, $y' = dy$, $y = x$, $y = -x$, where a and d are constants, and (x, y) and (x', y') are points before and after the transform, respectively. We prepared the 11 points x = (-1.0, -0.8, ..., 1.0).

First, we examined the learning, forward estimation, and inverse estimation of the symmetric expansion/compression transform by setting parameters $a = d = 0.5$. The forward mapping vector by forward estimation and the inverse mapping vector by inverse estimation are shown in Figs. 10 (a) and (b), respectively. Based on these results, it was confirmed that the transform vectors were correctly obtained. In Fig. 11, we show the results of the forward and inverse estimation of the expansion/compression transform of a circle, whose radius and center are 0.5 and the origin, respectively. Based on these results, it was confirmed that the forward and inverse estimations were correctly conducted. In addition, Fig. 12 shows the results of the forward and inverse estimation of the expansion/compression of the Gaussian function. Based on these results, it was found that the expansion/compression is correctly conducted. Therefore, it was shown that a complex-valued neural network can realize the forward and inverse transforms of expansion/compression.

Next, we examined the learning, forward estimation, and inverse estimation of an asymmetric expansion/compression transform by setting parameters a = 1.0, d = 0.2. Fig. 13 shows the forward mapping vector and the inverse mapping vector obtained by learning. Based on these results, it was found that the expansion/compression was applied only along the y-axis. Fig. 14 shows the results of the forward and inverse estimation of the expansion/compression transform of a circle, whose radius and center are 0.5 and the origin, respectively. Based on these results, it was found that the forward and inverse estimations of the expansion/compression were correctly shown to be an oval distribution of the estimated points. In addition, Fig. 15 shows the results of the forward and inverse estimation of the expansion/compression of the Gaussian function. Therefore, it was shown that a complex-valued neural network can realize the forward and inverse transforms of the asymmetric expansion/compression.

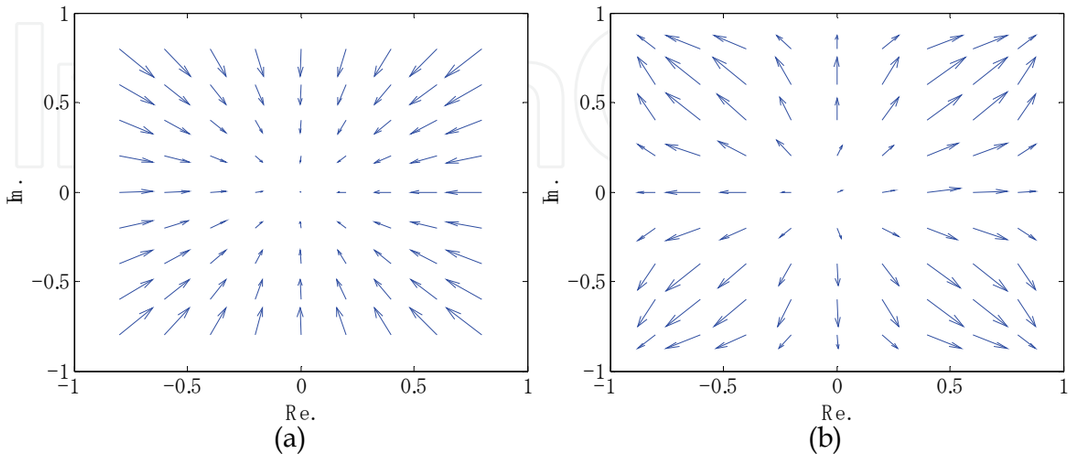
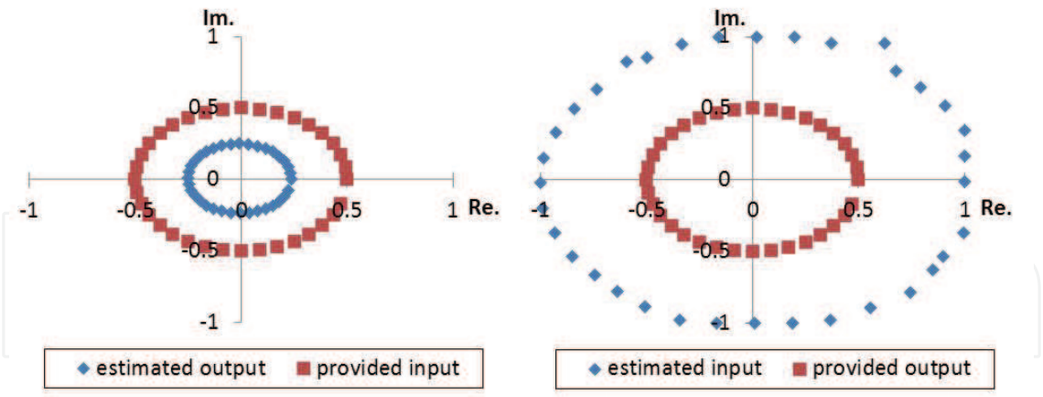
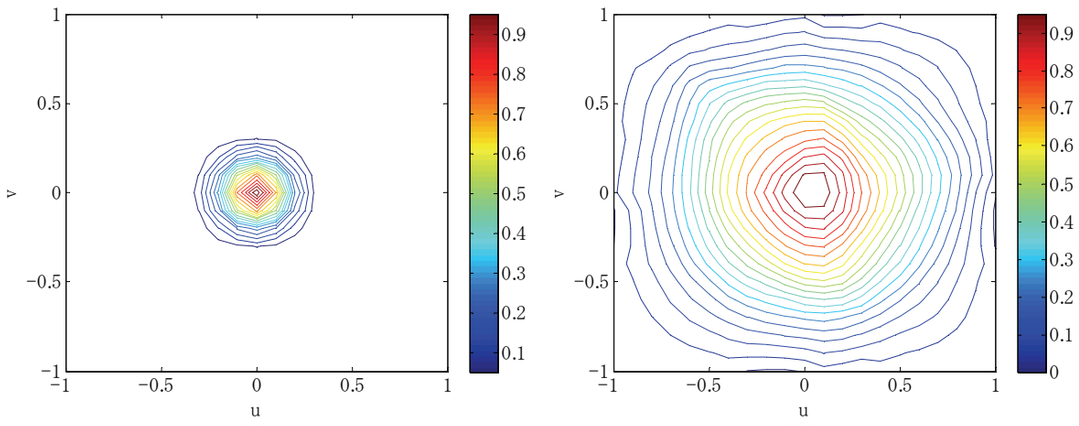


Fig. 10. Transform vector obtained by learning of the symmetric expansion for (a) forward estimation and (b) inverse estimation.



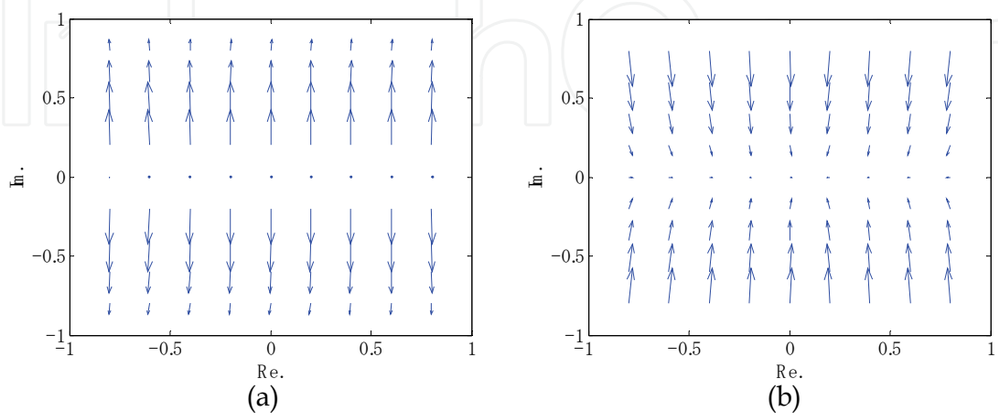
(a) (b)

Fig. 11. Results of symmetric expansion of a circle by (a) forward estimation and (b) inverse estimation.



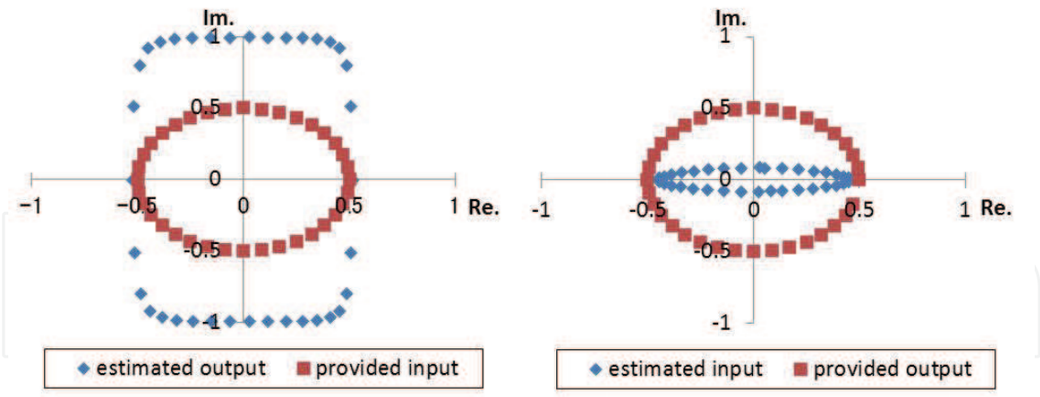
(a) (b)

Fig. 12. Results of symmetric expansion of a Gaussian function by (a) forward estimation and (b) inverse estimation.



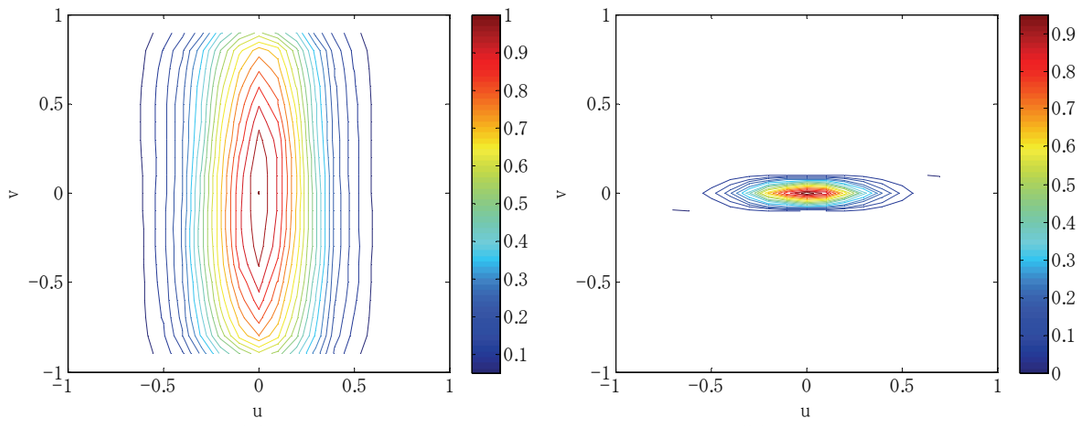
(a) (b)

Fig. 13. Transform vector obtained by the learning of asymmetric expansion with (a) forward estimation and (b) inverse estimation.



(a) (b)

Fig. 14. Results of the asymmetric expansion of a circle by (a) forward estimation and (b) inverse estimation.



(a) (b)

Fig. 15. Results of asymmetric expansion of a Gaussian function by (a) forward estimation and (b) inverse estimation.

4.2 Rotation transform

We examined the learning, forward estimation, and inverse estimation of the rotation transform on a complex plane with complex-valued neural networks. As learning data, we provided 22 pairs of data that satisfied $y' = x\sin\theta + y\cos\theta$, $y = x$, $y = -x$, where (x, y) and (x', y') are the points before and after the transform, respectively. We prepared the 11 points x = (-0.5, -0.4, ..., 0.5).

We examined the learning, forward estimation, and inverse estimation of the rotation transform by setting parameter θ = 45°. Fig. 16 shows the forward mapping vector and the inverse mapping vector obtained by learning. Based on these results, it was confirmed that the transform vectors were correctly obtained. Fig. 17 shows the results of the forward and inverse estimation of the rotation transform of an ellipse whose major axis, minor axis, and center were 0.5, 0.125, and the origin, respectively. Based on these results, it was confirmed

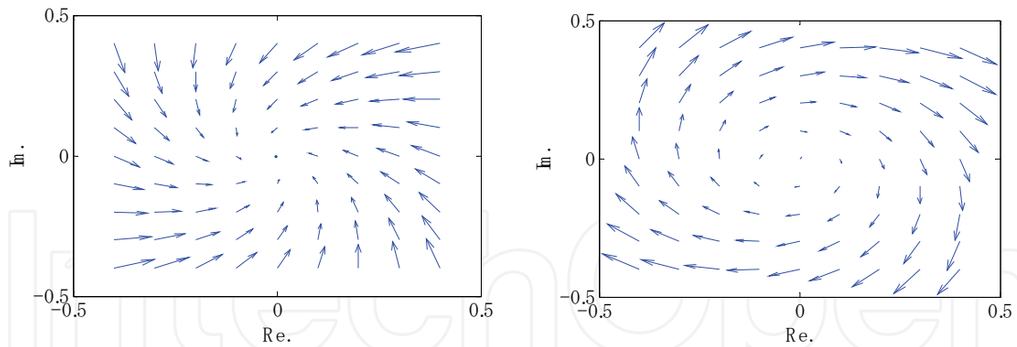


Fig. 16. Transform vector obtained by learning of the rotation with (a) forward estimation and (b) inverse estimation.

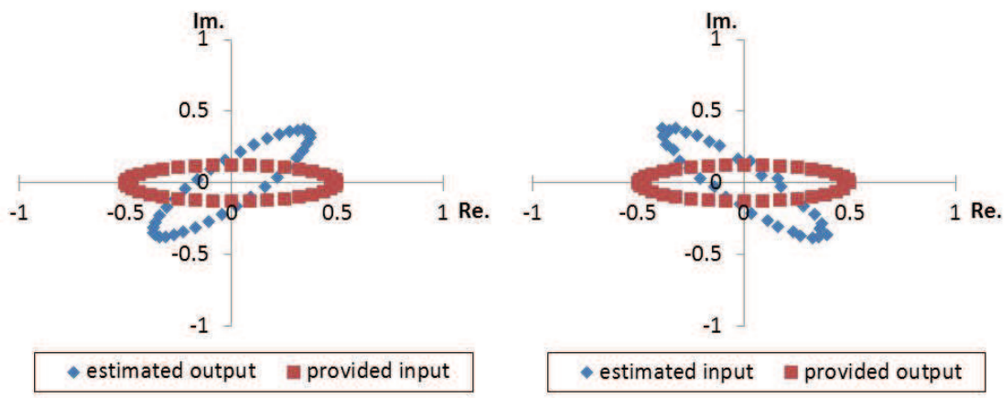


Fig. 17. Results of the rotation of an ellipse by (a) forward estimation and (b) inverse estimation.

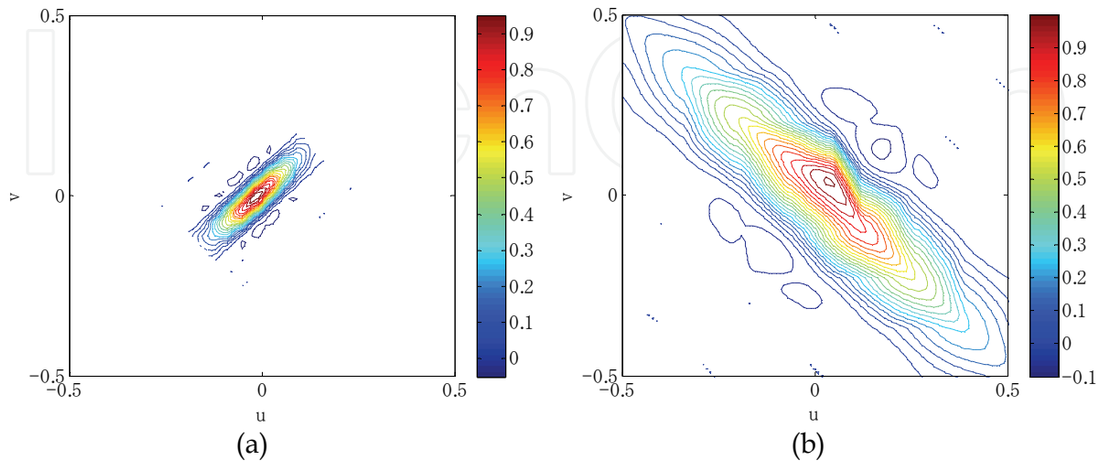


Fig. 18. Results of the rotation of an elliptic Gaussian function by (a) forward estimation and (b) inverse estimation.

that the forward and inverse estimations were correctly conducted. In addition, Fig. 18 shows the results of the forward and inverse estimation of the rotation of an elliptic Gaussian function. Based on these results, it was found that the rotation was correctly conducted. Therefore, it was shown that a complex-valued neural network can realize the forward and inverse transforms of the rotation.

4.3 Projective transform

We examined the learning, forward estimation, and inverse estimation of a projective transform on a complex plane with complex-valued neural networks. As learning data, we prepared 22 pair of data on $y = x$ or $y = -x$ that satisfied equation (19), where (x, y) and (x', y') were the points before and after the transform, respectively. We prepared the 11 points x = (-0.5, -0.4, ..., 0.5).

First, we examined the learning, forward estimation, and inverse estimation of a projective transform by setting the parameters $a_{11} = 3$, $a_{12} = 0$, $a_{13} = 0$, $a_{21} = 0$, $a_{22} = 3$, $a_{23} = 0$, $a_{31} = 0$, $a_{32} = 6$, and $a_{33} = 6$. Fig. 19 shows the forward mapping vector and the inverse mapping vector

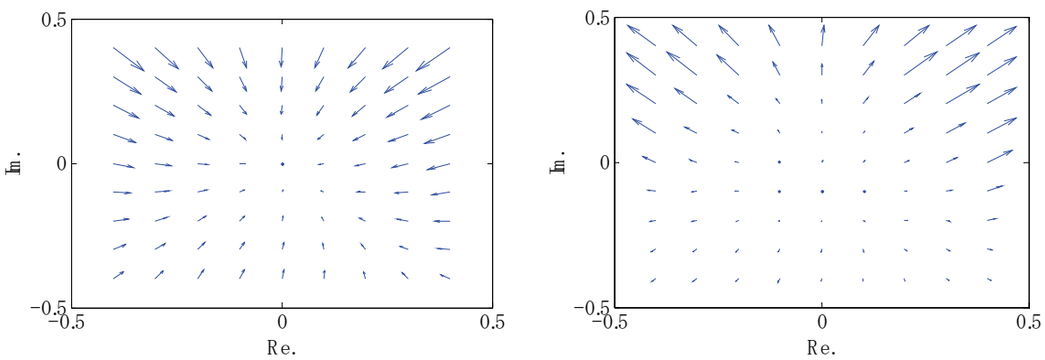


Fig. 19. Transform vector obtained by learning a projective transform with (a) forward estimation and (b) inverse estimation.

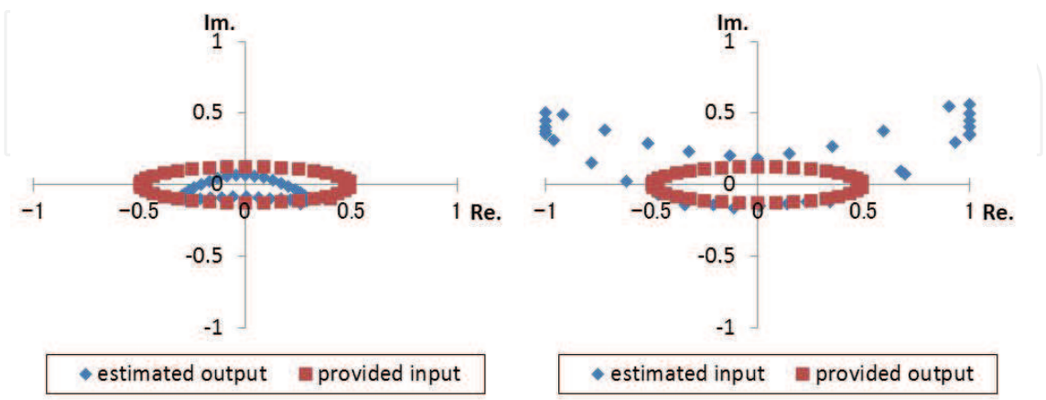


Fig. 20. Results of the projective transform of a circle by (a) forward estimation and (b) inverse estimation.

obtained by learning. Based on these results, it was confirmed that the transform vectors were correctly obtained. Fig. 20 shows the results of the forward and inverse estimation of the rotation transform of an ellipse whose major axis, minor axis, and center were 0.5, 0.125, and the origin, respectively. Based on these results, it was confirmed that the forward and inverse estimations were correctly conducted. In addition, Fig. 21 shows the results of the forward and inverse estimation of the rotation of an elliptic Gaussian function. Based on these results, it was found that the rotation was correctly conducted. Therefore, it was shown that a complex-valued neural network can realize the forward and inverse transforms of a projective transform.

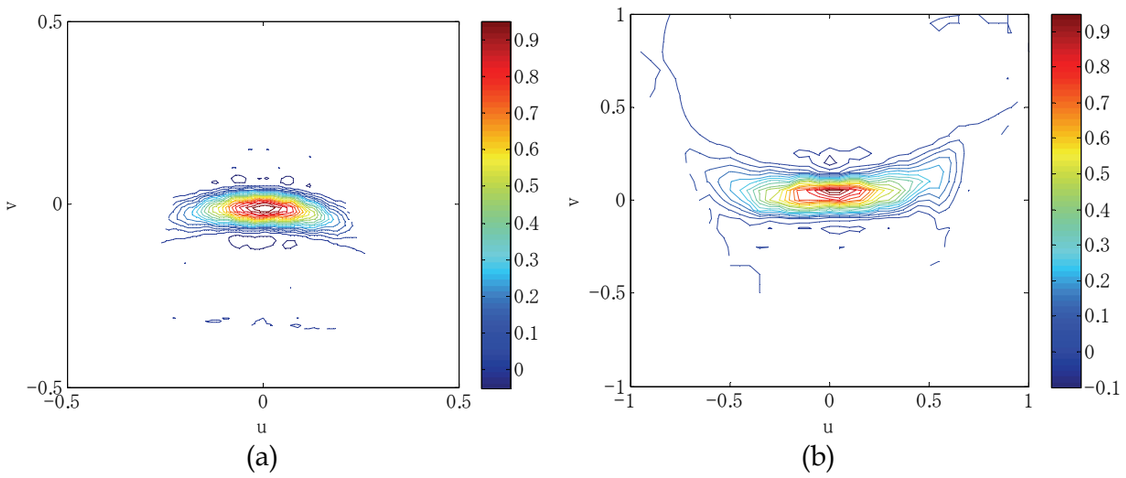


Fig. 21. Results of the projective transform of an elliptic Gaussian function by (a) forward estimation and (b) inverse estimation.

4.4 Filtering images using transformed filters

We performed image filtering using the transformed filter estimated in the previous simulations and examined the results. We used four transformed filters: an expanded Gaussian low-pass filter with forward estimation, an expanded Gaussian low-pass filter with inverse estimation, a rotated elliptic Gaussian filter with forward estimation, and a rotated elliptic Gaussian filter with inverse estimation. The transformed image spectrum $I'(u, v)$ can be described as the product of the transfer function $H(u, v)$ and the original image spectrum $I(u, v)$; that is,

$$I'(u,v) = H(u,v)I(u,v) \tag{22}$$

The standard image on which the filtering was performed is shown in Fig. 22(a). Fig. 22(b) shows simple line segments that indicate the orientation selectivity of the rotated elliptic

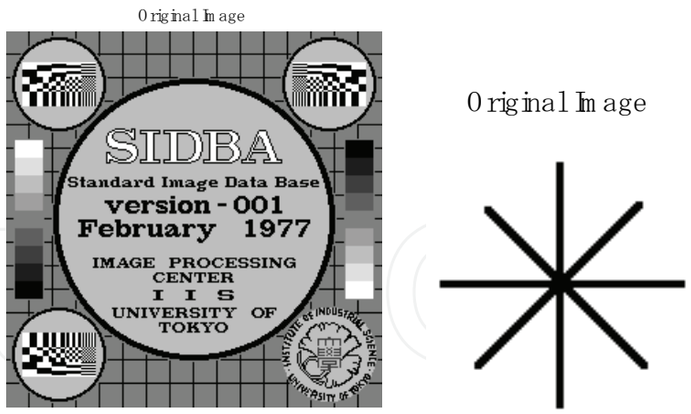
Gaussian filter. The results for the standard image are shown in Fig. 23. Fig. 24 shows a magnification of the upper-left part of Fig. 23. Part (a) in Figs. 23 and 24 shows the standard image filtered using the expanded Gaussian low-pass filter with forward estimation; part (b) shows the case for expanded Gaussian low-pass filter with inverse estimation. In Fig. 23 (c) and (d), a striped pattern flowing from upper left to lower right and from upper right to lower left, respectively, is observed; this indicates that orientation selection was performed. To clearly demonstrate the effects of the orientation selectivity, the simple line segments were filtered by transformed elliptic Gaussian filter, as shown in Fig. 25. It was found that the line segments at 45° and -45° were selectively extracted, as shown in Fig. 25 (a) and (b), respectively. This result confirmed that directional selectivity had been achieved.

The above results confirm that the low-pass filter and orientation selection filter can be correctly realized by transforming image filters using a neural network. We can obtain a low-pass filter and an orientation selection filter having arbitrary characteristics by appropriate scaling and rotation of a Gaussian filter. An advantage of the neural network based method is that arbitrary filter characteristics can be obtained by learning of the input/output relation.

The advantage of the transformation of the image filter by a neural network is the adaptive composition of the filter by learning. As shown in the above simulation, the use of a neural network enables the easy realization of not only linear transformations, such as scaling and rotation, but also a nonlinear transformation, namely, projective transformation. Because neural networks can obtain a transformation only by a mapping between input and output, any transformation is realizable if input and output are known. Moreover, the use of neural networks enables the realization of an inverse filter by a network inversion method, which can be applied to areas such as image restoration and bidirectional image processing. The advantage of transformation by complex-valued neural networks is that the mapping on a plane, especially one that has undergone rotation, can be expressed well. Moreover, complex-valued neural networks afford lower computational complexity than conventional real-valued neural networks because, in the former, two-dimensional information can be expressed by a neuron.

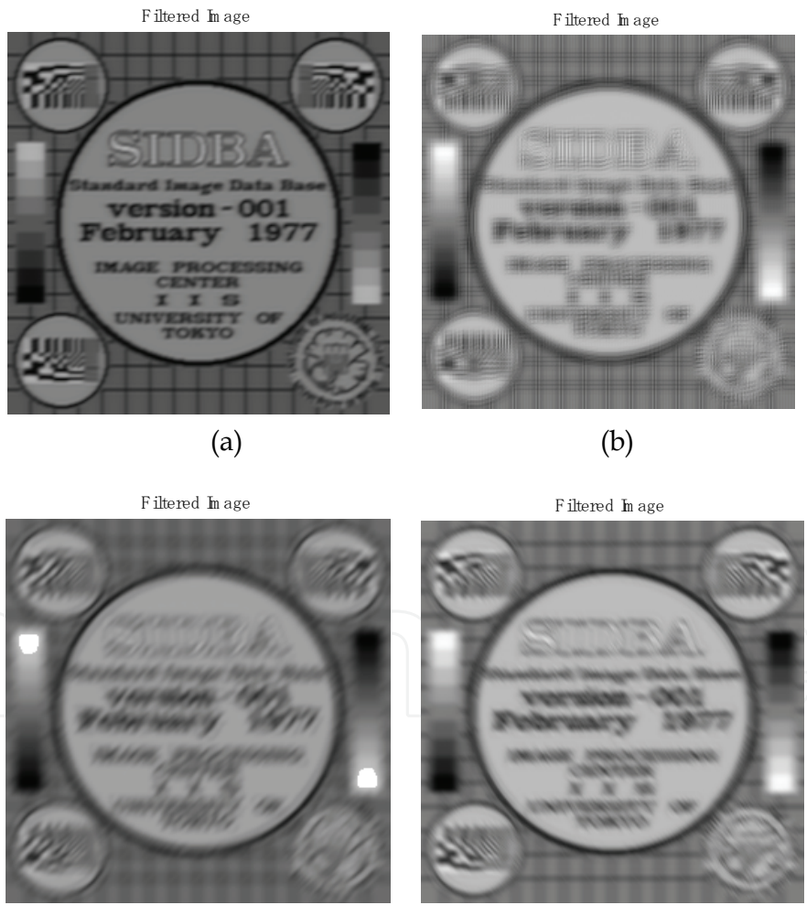
In machine vision, the advantage of filter transformation by the proposed neural network is as follows. Neural networks are often used as part of the recognition system in a machine vision system. The filter transformation proposed in this work corresponds to the pre-processing stage in machine vision. Because neural network based pre-processing is compatible with a neural network based recognition system, we will be able to integrate the pre-processing and recognition system when designing a machine vision system. Furthermore, the model of a recognition neural network, such as the neocognitron (Fukushima, 1988), includes an orientation selection cell. Therefore, we expect that the proposed method can be directly applied to such a model.

The experimental results indicate that the proposed filter afforded most of the expected advantages. The disadvantage of the proposed neural network based method is that its computational complexity is slightly high. However, it is expected that this can be resolved by further refinement of the computer, system, and calculation method.



(a) (b)

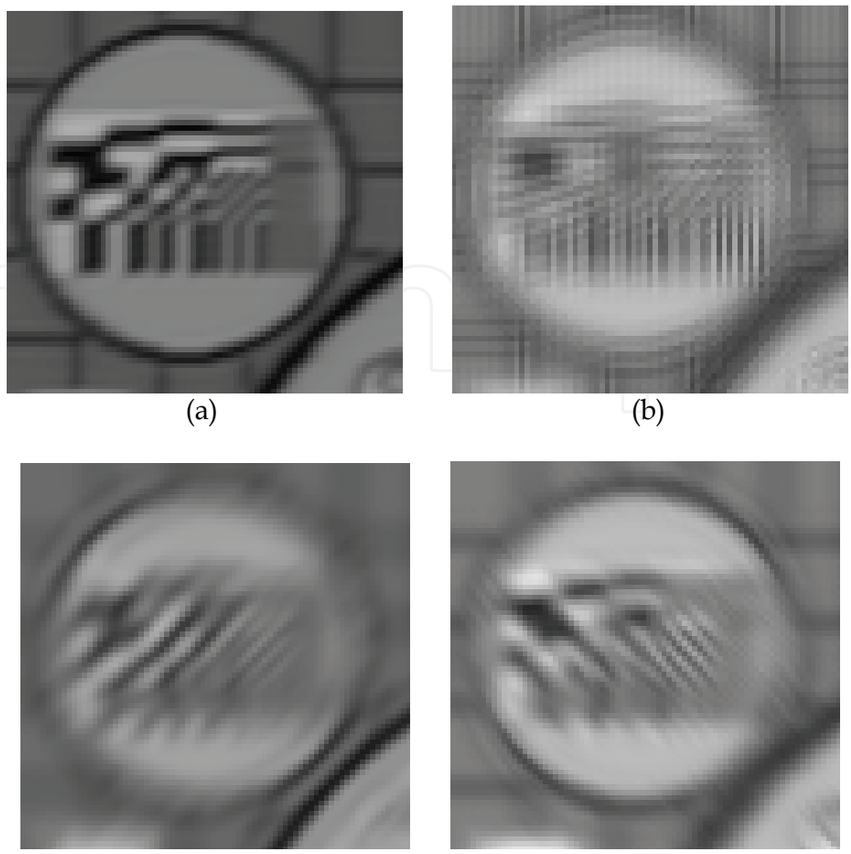
Fig. 22. Original images used in the simulation: (a) standard image (b) simple line segments.



(a) (b)

(c) (d)

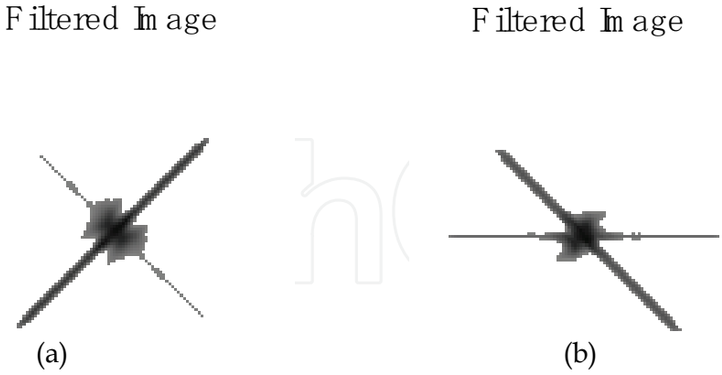
Fig. 23. Images filtered using (a) an expanded Gaussian low-pass filter with forward estimation, (b) an expanded Gaussian low-pass filter with inverse estimation, (c) a rotated elliptic Gaussian filter with forward estimation, and (d) a rotated elliptic Gaussian filter with inverse estimation.



(a) (b)

(c) (d)

Fig. 24. Magnified images of the upper left part of Fig. 23.



(a) (b)

Fig. 25. Image of simple line segments filtered using (a) a rotated elliptic Gaussian filter with forward estimation, and (b) a rotated elliptic Gaussian filter with inverse estimation.

5. Conclusion

In this study, we showed the mapping ability of a complex-valued neural network and proposed its use in the transform of an image filter. We introduced a complex-valued multilayer neural network for solving forward problems and a complex-valued network

inversion for solving inverse problems and explained each principle of operation in detail. Moreover, to demonstrate the mapping ability, we handled complex mapping problems, i.e., the linear geometric transforms, namely, expansion/compression, rotation, and projective transforms. We confirmed the estimation procedure by simulation. In addition, we applied this mapping capacity to the transform of an image filter, and we demonstrated the geometric transform of various filters, such as the Gaussian filter. Moreover, we examined image filtering with a filter that was transformed using a complex-valued neural network and confirmed its effectiveness.

As the future work, we will apply complex-valued neural networks to nonlinear transforms of various image filters. Moreover, we will introduce processing with complex-valued neural networks into an actual robot vision system based on the results of this study.

6. References

Benvenuto, N. & Piazza, F. (1992). On the Complex Backpropagation Algorithm, *IEEE Trans. on Signal Processing*, vol.40, no.4, pp.967–969.

Du, K. L., Lai, A. K. Y., Cheng, K. K. M. & Swamy, M. N. S. (2002). Neural Methods for Antenna Array Signal Processing: A Review, *Signal Processing,* vol.82, pp.547–561.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition, *Neural Networks*, vol 1, pp. 119-130.

Gonzalez, R. C. & Woods, E. W. (2010). *Digital Image Processing*, Pearson Education International, ISBN: 978-0132345637, New Jersey.

Groetsch, C. W. (1993). *Inverse Problems in the Mathematical Sciences*, Friedr. Vieweg and Sohn Verlags. mbH, ISBN: 978-3528065454, Stuttgart.

Hara, T. & Hirose, A. (2004). Plastic Mine Detecting Radar System Using Complex-Valued Self-Organizing Map That Deal with Multiple-Frequency Interferometric Images, *Neural Networks*, vol.17, no.8–9, pp.1201–1210.

Hirose, A. & Hara, T. (2003). Complex-Valued Self-Organizing Map Dealing with Multi-Frequency Interferometeric Data for Radar Imaging Systems, *Proc. of WSOM 2003*, pp. 255–260.

Hirose, A. (2006). *Complex-Valued Neural Networks*, Springer, ISBN: 978-3540334569, New York.

Linden, A. & Kindermann, J. (1989). Inversion of Multilayer Nets, *Proc. Int. Joint Conf. on Neural Networks*, pp.425–430.

Nemoto, I. & Kubono, M. (1996). Complex Associative Memory, *Neural Networks*, vol.9, pp. 253–261.

Nishikawa, I. & Kuroe, Y. (2004). Dynamics of Complex-Valued Neural Networks and Its Relation to a Phase Oscillator System, *Proc. of ICONIP 2004*, pp.122–129.

Nitta, T. (1997). An Extension of the Backpropagation Algorithm to Complex Numbers, *Neural Networks*, vol.10, no.8, pp.1392–1415.

Ogawa, T. & Kanada, H. (2010). Solution for Ill-Posed Inverse Kinematics of Robot Arm by Network Inversion, *Journal of Robotics, Hindawi Publishing*.

Ogawa, T. (2009). *Complex-Valued Neural Network and Inverse Problems*, in *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*, (Dr. Nitta, T. ed.), IGI-Global, ISBN 978-160566214-5, New York, Chapter 2, pp.27-55.

Pratt, W. K. (2007). *Digital Image Processing*, John Wiley & Sons, ISBN: 978-0471767770, New Jersey.

Valova, I., Kameyama, K. & Kosugi, Y. (1995). Image Decomposition by Answer-in-Weights Neural Network, *IEICE Trans. on Information and Systems*, vol.E78-D-9, pp.1221–1224.

Human-Centric Machine Vision
Edited by Dr. Fabio Solari

ISBN 978-953-51-0563-3
Hard cover, 180 pages
Publisher InTech
Published online 02, May, 2012
Published in print edition May, 2012

Recently, the algorithms for the processing of the visual information have greatly evolved, providing efficient and effective solutions to cope with the variability and the complexity of real-world environments. These achievements yield to the development of Machine Vision systems that overcome the typical industrial applications, where the environments are controlled and the tasks are very specific, towards the use of innovative solutions to face with everyday needs of people. The Human-Centric Machine Vision can help to solve the problems raised by the needs of our society, e.g. security and safety, health care, medical imaging, and human machine interface. In such applications it is necessary to handle changing, unpredictable and complex situations, and to take care of the presence of humans.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen